Uniwersytet Jagielloński Instytut Informatyki KIS Katedra Informatyki i Stosowanej	Metody programowania 2023/2024 Konwersje: ONP ⟺ INF	P_03

Opis

Napisz program w Javie, który będzie realizował następujące operacje:

1. Konwertuje wyrażenia arytmetyczne i instrukcje przypisania z notacji INF do ONP.
2. Konwertuje wyrażenia arytmetyczne i instrukcje przypisania z ONP do notacji INF, zawierającej minimalną liczbę nawiasów, gwarantującą taką kolejność obliczeń jak w wyrażeniu ONP.

Przy czym instrukcja przypisania ma postać: wyrażenie_arytm1 = wyrażenie_arytm2
natomiast wyrażenia arytmetyczne składają się z poniższych symboli:

a. nawiasy: (,) - tylko w notacji INF
b. operandy: małe litery alfabetu angielskiego
c. operatory o priorytetach i łączności podanej poniżej

operator	priorytet	łączność	opis operatora
( )	najwyższy	lewostronna	nawiasy
!, ~		prawostronna	negacja , - unarny
^		prawostronna	potęgowania
*, /, %		lewostronna	multiplikatywny
+, -		lewostronna	addytywny
<, >		lewostronna	relacje < i >
&		lewostronna	koniunkcja
I		lewostronna	alternatywa
=	najniższy	prawostronna	przypisanie

Ponadto badana jest poprawność wyrażeń w postaci INF, opisana w następnym punkcie.

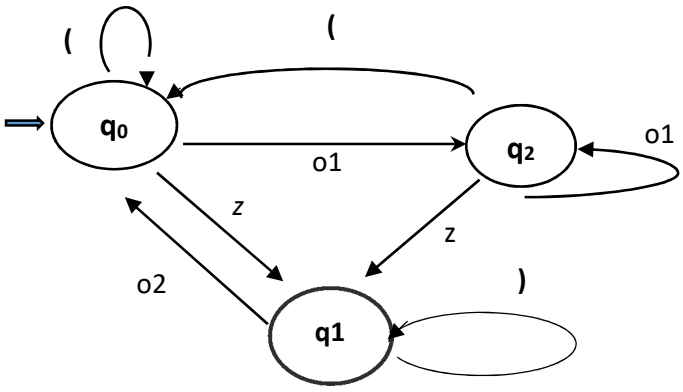
W przypadku wyrażenia w ONP, po zignorowaniu znaków, które nie mogą w nim występować, wyrażenie uważamy za poprawne jeśli jest wykonalne.

Poprawność wyrażeń arytmetycznych w postaci infiksowej (INF)

Badanie poprawności wyrażeń arytmetycznych po usunięciu zbędnych znaków składa się z dwóch kroków:

1) Sprawdzenie, czy wyrażenie jest akceptowane przez poniższy automat skończony,
   $A=(Q, \Sigma, \delta, q_0, F)$, gdzie
   $Q=\{q_0, q_1, q_2\}$ – zbiór stanów
   Σ = { z, o1, o2, (,) } – symbole, które mogą wystąpić w wyrażeniu, przy czym:
   z - operand - zmienna (pojedyncza litera),
   o1 - operatory jednoargumentowe { ~ , !)
   o2 - operatory dwuargumentowe { ^ * / % + - < > = & | }
   ( ) - nawiasy
   δ - funkcja przejścia automatu, którą definiuje poniższy graf
   q_0 - stan początkowy, $F = \{q_1\}$ – zbiór stanów końcowych

   Możemy przyjąć że automat rozpoczyna stanie q_0 - analizę wyrażenia w postaci INF od pierwszego symbolu wyrażenia.
   Jeśli automat po przeczytaniu wyrażenia znajdzie się w stanie q_1 wówczas akceptuje wyrażenie.



2) Aby wyrażenie było poprawne trzeba jeszcze sprawdzić, czy nie występują w wyrażeniu następujące przypadki:
   a. niesparowane nawiasy lub ich zła kolejność, np. a)+(a , a+b)+(c , (a)), () a+ b
   b. niezgodność liczby operatorów i operandów, np. –/+ b*c , a~+b, a+b~

Wejście

Dane do programu wczytywane są ze standardowego wejścia zgodnie z poniższą specyfikacją. Pierwsza linia wejścia zawiera liczbę całkowitą *z,* oznaczającą liczbę linii zawierających wyrażenia arytmetyczne, których opisy występują kolejno po sobie.

Każda linia zawiera co najmniej 6 znaków i nie przekracza 256 znaków, może mieć jedną z dwóch postaci:

INF: wyrażenie arytmetyczne, zapisane w notacji infiksowej,

ONP: wyrażenie arytmetyczne, zapisane w notacji ONP

Przy czym wyrażenia mogą zawierać dowolne znaki. Program ignoruje znaki niewystępujące w wyrażeniach arytmetycznych, w tym spacje oraz sprawdza poprawność wyrażeń.
Ostatnia linia każdego zestawu zakończona jest znakiem '\n'.

Wyjście

- Wyrażenie poprzedzone na wejściu napisem "**`INF: `**" musi być na wyjściu poprzedzone napisem "**`ONP: `**" i analogicznie wyrażenie poprzedzone na wejściu napisem **"`ONP: `"** musi być na wyjściu poprzedzone napisem **"`INF: `"**
  W przypadku błędnego wyrażenia, na wyjściu, zamiast skonwertowanego wyrażenia pojawi napis **`error`**.
- W przypadku konwersji wyrażenia w ONP do w INF, wyrażenie w INF musi zawierać minimalną liczbę nawiasów, gwarantującą podczas obliczania taką kolejność operacji (uwzględniając typ łączności i priorytety operatorów) jak w wyrażeniu ONP,
  np. **`ONP: abc**`** zostanie przekształcone do **`INF: a * ( b * c )`**
- W przypadku wyrażenia w notacji **INF**, np. **`INF: ( a,+ b)/..[c3`** program pozostawia jedynie: **`(a+b)/c`** pozostałe znaki, w tym spacje – odrzuca, dodatkowo sprawdza poprawność wyrażenia, po czym dokonuje konwersji, wypisując na wyjściu: **`ONP: a b + c /`**
- W przypadku wyrażeń w notacji **ONP**, np. **`ONP: ( a,b,.).c;-,*`** program pozostawia jedynie: **`abc-*`** dodatkowo sprawdza, czy wyrażenie jest poprawne, po czym dokonuje konwersji, wypisując na wyjściu: **`INF: a*(b-c)`**
- Wszystkie elementy wyrażeń na wyjściu są poprzedzone pojedynczą spacją.

Wymagania implementacyjne

Jak poprzednio jedynym dozwolonym importem jest obsługa wczytywania z klawiatury, to jest: *import java.util.Scanner*, tym samym klasę stosu należy zaimplementować samodzielnie.

Ze względu na efektywność programów analiza poprawności wyrażenia (kroki 1 i 2) powinna być wykonywana w głównej pętli algorytmu konwersji.

Przykład danych

wejście:	Wyjście:
10	
INF: x = !(c>~a) & ~c<b	ONP: x c a ~ > ! c ~ b < & =
INF: x = !(c>a & c<b)	ONP: x c a > c b < & ! =
ONP: x c a > c b < & ! =	INF: x = ! (c > a & c < b)
INF: x = !!(c>a | c<b)	ONP: x c a > c b < | ! ! =
INF: x = !(!c>a & !c<b)	ONP: x c ! a > c ! b < & ! =
ONP: xca>cb<|!!=	INF: x = ! ! (c > a | c < b)
INF: x = i > k | i<n & n > k	ONP: x i k > i n < n k > & | =
INF: x = ~ ~ ~ a	ONP: x a ~ ~ ~ =
INF: a ~ + b	ONP: error
INF: a + b ~	ONP: error